Session Fixation

Session fixation is a pretty small but common vulnerability.

A common way to handle the fact that HTTP is a stateless protocol is you store cookies in the users browser, and then have that cookie send to the web server on each subsequent request. This way the web server can know that the user has visited the website before. So when a user logs in to a web application a cookie for that session is usually created, in order for the web-server to know that the session is active.

Session fixation happens when the session-identifier (in this case the cookie) is setbefore the user has authenticated itself (which is usually done with a simple username/password login), and then not changed when the user authenticates itself.

For example, let's say you want to log in to a web application. When you first visit the site the following cookie is set:

```
SessionID=123ad76dab97b23ba8d76a
```

You then authenticate with your username and password and make a successful login. But the SessionID-cookie does not change. Then you have a session fixation vulnerability on your hands. Because this means that if an attacker can set the SessionID-cookie to a value the attacker knows it will then know the SessionID-cookie once the user actually authenticates.

How to set the cookie?

In GET request - if the session-token is sent in the URL of a GET-request the attacker can simply send a link which contains the attacker-controlled session-token.

XSS - If the attacker has also found a XSS vulnerability she can use it to set the cookie. This can of course be mitigated by setting the HttpOnly attribute to the cookie.

META-tag - If the attacker has the ability to inject html-code she can use the META-tag to set the cookie.

```
http://website.kon/<meta http-equiv=Set-Cookie content=”sessionid=abcd”>
```

MITM - By being MITM the attacker can set the cookie.